Title

stata.com

vl rebuild — Rebuild variable lists

Description

`vl rebuild` restores system-defined and user-defined variable lists. After loading a dataset with `use`, run `vl rebuild`.

After using `merge` or `append`, run `vl rebuild` to merge variable lists. You only need to run `vl rebuild` when the using dataset has variable lists.

After dropping variables with `drop`, run `vl rebuild` to remove the dropped variables from all variable lists.

After modifying variable lists with `vl modify` or `vl move`, run `vl rebuild` to update variable lists created by `vl substitute`.

And if you are confused, know that it never hurts to run `vl rebuild`.

For an introduction to the `vl` commands, see [D] **vl**.

Quick start

Restore variable lists after loading a dataset with `use`

```
vl rebuild
```

After running `merge` when the using dataset has variable lists, merge its variable lists into those in the master dataset

```
vl rebuild
```

After dropping variables with `drop`, remove the dropped variables from all variable lists

```
vl rebuild
```

Update a variable list created by `vl substitute` after modifying any of its component variable lists

```
vl rebuild
```

Syntax

```
vl rebuild
```

`collect` is allowed; see [U] **11.1.10 Prefix commands**.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Reloading datasets

System-defined and user-defined variable lists are saved with the dataset. However, they are not automatically restored when you reload the data. Just type `vl rebuild` to restore them.

```
. use ...
. vl rebuild
```

Merging datasets

Another time when `vl rebuild` is needed is when a `merge` is done and the `using` dataset has variable lists.

```
. merge ... using filename
. vl rebuild
```

Only when *filename* has variable lists is it necessary to run `vl rebuild`. When both the master dataset in memory and *filename* have variable lists, `vl rebuild` merges them. When the master dataset has variable lists but *filename* does not, there is no need to run `vl rebuild`. However, running `vl rebuild` is always harmless.

Dropping variables

When you drop variables from the data in memory using `drop`, the dropped variables are not automatically removed from variable lists. They can be explicitly removed by using `vl drop`.

```
. drop varlist
. vl drop (varlist)
```

Instead of running `vl drop` with the list of variables that were dropped, you can simply type

```
. vl rebuild
```

It will do the same thing, and you do not have to remember the names of the variables that were dropped.

If you drop or add observations or change any of the values of variables in variable lists, `vl rebuild` does not update the stored variable statistics, namely, the number of levels, the minimum and maximum values, and the number of nonmissing observations. If you want to update these statistics without changing the system-defined classifications, type

```
. vl set, update
```

If you want to update the statistics and redo the system-defined classifications for all variables, type

```
. vl set, clear
```

See [D] **vl set**.

vl substitute and vl rebuild

`vl rebuild` has another important use. It will update variable lists created by `vl substitute`.

For example, we created two user-defined variable lists:

```
. vl create myfactors = (x1 x2 x3)
. vl create mycontinuous = (c1 c2 c3 c4 c5)
```

Then we created a variable list using factor-variable operators:

```
. vl substitute myinteraction = i.myfactors##c.mycontinuous
```

If we modify `mycontinuous`,

```
. vl modify mycontinuous = mycontinuous - (c3)
```

then the global macro `$myinteraction` for the variable list `myinteraction` remains unchanged.

Running

```
. vl rebuild
```

updates the global macro `$myinteraction`.

Again, if you make any changes to your data or to your variable lists, and you want to make sure everything is set properly and up to date, just type

```
. vl rebuild
```

Characteristics

Advanced Stata users will likely guess how variable lists and variable statistics are stored with the dataset. They are stored as characteristics. If you want to see them, type

```
. char list
```

See [P] **char**.

Stored results

`vl rebuild` stores the following in `r()`:

Scalars

`r(k_system)`	number of variables in system-defined variable lists
`r(k_vlcategorical)`	number of variables in `vlcategorical`
`r(k_vlcontinuous)`	number of variables in `vlcontinuous`
`r(k_vluncertain)`	number of variables in `vluncertain`
`r(k_vlother)`	number of variables in `vlother`
`r(k_vldummy)`	number of variables in `vldummy` when defined
`r(k_user)`	number of variables in user-defined variable lists
`r(k_`*vlusername*`)`	number of variables in *vlusername*

Macros

`r(vlsysnames)`	names of system-defined variable lists
`r(vlusernames)`	names of user-defined variable lists

Also see

[D] **vl** — Manage variable lists

[D] **vl create** — Create and modify user-defined variable lists

[D] **vl drop** — Drop variable lists or variables from variable lists

[D] **vl list** — List contents of variable lists

[D] **vl set** — Set system-defined variable lists